

AD-A093 265

VIRGINIA POLYTECHNIC INST AND STATE UNIV WASHINGTON --ETC F/6 9/2
FINAL SCIENTIFIC REPORT.(U)

NOV 80 R J ORGASS

AFOSR-79-0021

UNCLASSIFIED

VPI/SU-TM-80-7

AFOSR-TR-80-1292

NL

1 34 1
0000



END

DATE
FILMED
1-8
DTIC



AFOSR-TR- 80 - 1292

(12)

EXTENSION DIVISION

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE
GRADUATE PROGRAM IN NORTHERN VIRGINIA

P. O. Box 17186
Washington, D. C. 20041
(703) 471-4600

LEVEL II

Final Scientific Report

Grant AFOSR-79-0021

Richard J. Orgass

Technical Memorandum No. 80-7

November 22, 1980

DTIC
ELECTE
DEC 23 1980
S F D

Submitted to

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Building 410

Bolling Air Force Base

Washington, D.C. 20332

DOC FILE COPY

Approved for public release
distribution unlimited

© 1980 by Richard J. Orgass

General permission to republish, but not for profit, all or part of this report is granted, provided that the copyright notice is given and that reference is made to the publication (Technical Memorandum No. 80-7, Department of Computer Science, Graduate Program in Northern Virginia, Virginia Polytechnic Institute and State University), to its date of issue and to the fact that reprinting privileges were granted by the author.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER (18) AFOSR/TR- 80-1292	2. GOVT ACCESSION NO. AD-A093 265	3. RECIPIENT'S CATALOG NUMBER (9) Technical Report	
4. TITLE (and Subtitle) (6) FINAL SCIENTIFIC REPORT		5. TYPE OF REPORT & PERIOD COVERED Final	
		6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) (10) Richard J. Orgass		8. CONTRACT OR GRANT NUMBER(s) (15) AFOSR-79-0021	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Computer Science Virginia Polytechnic Institute and State University		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F (17) 2304A5	
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, D. C. 20332		12. REPORT DATE (11) 22 November 1980	
		13. NUMBER OF PAGES 11	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (14) VPI/SU-IM-89.1		15. SECURITY CLASS. (of this report) UNCLASSIFIED	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release, distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The work performed under this grant can be divided into three time periods. From October 1979 to January 1980 the main emphasis of grant work was bringing software that was reliably working on a DECsystem-10 into reliable operation on the Virginia TECH VM/CMS system. This work is best viewed as preparation for the substantial work of the grant. 4111 731			

Table of Contents

1. Overview	1
2. APL Semantics and Implementation	3
3. Verification of APL Programs	4
4. File System Development	5
5. Other Software Development	8
6. Publications	9
7. Personnel	10
8. Publications	11

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

Final Scientific Report

Grant AFOSR-79-0021

1. Overview

The work performed under this grant can be divided into three time periods. From October 1979 to January 1980, the main emphasis of grant work was bringing software that was reliably working on a DECsystem-10 into reliable operation on the Virginia Tech VM/CMS system. This work is best viewed as preparation for the substantial work of the grant.

From January 1980 until the end of April 1980, ^{also} work was directed toward achieving two objectives: (1) Modify an incremental program verification system to accept APL programs. The main work here was to isolate code in the verifier that is specific to the syntax of the programming language in which the programs were written in a single SMULA class so that it could be easily modified for different languages. (2) Completing the implementation of an APL expression evaluator based on a formal semantics of APL and verifying this implementation with respect to the formal semantics. During this period, the objective was to complete a verified implementation of an APL expression evaluator and to have an interactive verifier for APL expressions working correctly by about August 15, 1980. These two pieces of software were then to be used to conduct a number of experiments that would aid in the design of a verifier for all of APL. ←

At the end of April 1980, the Principal Investigator decided to leave Virginia Tech to join Xerox Corporation at the on August 31, 1980. This decision combined with a desire to complete a publishable piece of work motivated a change in the work plan and objectives for the remaining period of the grant.

If the planned pieces of work had been completed as expected, a significant step toward realizing the overall objectives of the grant would have been achieved but there would have been two partially completed pieces of work neither of which would have been sufficient for publication. In order to achieve publishable results during the time from April to August 1980, work on the verifier was set aside so that full attention could be devoted to completing the formal definition of APL and completing an APL implementation that is verified with respect to the formal definition. Such an implementation was completed and part of the verification exists as a draft manuscript and the essential details of the remainder of the verification are recorded in working notes. The Principal Investigator is continuing the work of preparing a manuscript that describes this work.

To support both the incremental verifier and the APL implementation, a substantial effort was devoted to implementing, in the VM/CMS environment, a usable file system for interactive programs. A manuscript, "Files in an Interactive Environment" describing some of this work has been submitted for publication in *Software Practice & Experience*.

A draft manuscript of a research monograph, "A Primitive Recursive Semantics and Implementation of APL" has been submitted for publication as a volume in the Springer-Verlag *Lecture Notes in Computer Science* series.

Available computing resources proved to be a significant obstacle to productive work on this grant. Most of the first fifteen months were devoted to bringing software that was

satisfactorily working on a DECsystem-10 into operation in the VM/CMS environment. The major problems may be summarized as follows.

In spite of the fact that SMULA implementations are closely controlled by the Norwegian Computing Center and that the semantics of the language is expected to be the same in all implementations, different implementations have varying numbers of system errors. It turned out to be the case that programs used in this research exposed many problems in the IBM SMULA implementation.

The VM/CMS system is nominally an interactive system but it lacks many of the capabilities that one expects in a modern interactive computing system. Since these capabilities are essential for the work of the grant, it was necessary to devote considerable effort to creating a suitable working environment in VM/CMS. Some of this work is discussed in the paper described above.

Until the end of October 1979, this research work also suffered from the lack of reliable data communications between the Northern Virginia Graduate Center (the location of the principal investigator) and the Virginia Tech main campus in Blacksburg. Once reasonably reliable data communications became available (first with foreign exchange lines and later with statistical multiplexors on a dedicated wideband line), the work on software development proceeded much more quickly.

The specific pieces of software that were constructed with the support of this grant are:

- (1) An implementation of APL that includes user defined functions and operators but which relies on APL definitions of some primitive functions and operators was completed. The implementation supports file input and output but does not include facilities for saving workspaces in binary form.
- (2) An interactive incremental verifier which is substantially independent of the programming language that is used was brought into reliable operation under VM/CMS. Some work remains to be done to make the program completely independent of the programming language with which it is used.
- (3) An interactive file system with high quality terminal interfaces has been designed and partly implemented. A revision of this system to reduce the computing resources that it consumes was designed but the implementation was not completed.
- (4) A set of programs for parsing arbitrary SLR(1) languages and for manipulating the parse trees of such languages has been constructed. The implemented facilities include pattern matching on parse trees.
- (5) A VM/CMS implementation of RATFOR that is compatible with the UNIX and DEC-10 implementations of this preprocessor was completed. This preprocessor was transplanted from a DEC-10 so that it would be possible to transport software that is needed for the work of the grant from a DEC-10 to the Virginia Tech VM/CMS system.

2. APL Semantics and Implementation

Substantial effort was required to install the partial implementation of APL that was available when this grant began on the Virginia Tech VM/CMS system. Once this transfer was completed and the appropriate supporting software written, it was possible to complete an implementation of APL based on a formal definition of the language. A summary of some of the problems follows.

An unreasonable amount of time was devoted to tracking down apparent program errors that were found to be a consequence of the fact that certain characters (e.g., {, }, ~, |,]) have more than one EBCDIC code. Different system utilities provide different translations from ASCII to EBCDIC. All of the known character ambiguities can now be corrected by a program that was created for this purpose.

During the research period, five different SIMULA compilers were used. At the beginning of the grant work, MVS SIMULA, Version 6.00 was available at Virginia Tech. The partial implementation of APL that was available at the start of the grant exhibited a major error (as well as minor errors) in this compiler/run time system. When the problem was reported to the Norwegian Computing Center (SIMULA supplier), Version 6.02 of MVS SIMULA was provided to repair the problem.

This system made it possible to execute much of the interpreter but the terminal dialog was unacceptable as an approximation to the APL terminal interface. At this point, it appeared that a substantial assembly coding effort would be required to reasonably approximate the APL terminal interface. Just as a decision to write this assembly code was to be implemented, a version of IBM SIMULA specifically designed for CMS was announced by Imperial College of Science and Technology. This version has a few windows into the operating system and it appeared to be possible to provide a reasonable terminal interface with a minimum amount of assembly coding. Therefore, this system was acquired and installed at Virginia Tech. While this SIMULA system solved some problems, other system errors were present in this version.

The Principal Investigator was engaged in a regular correspondence with the Norwegian Computing Center during this period and a number of his suggestions for enhancements to IBM SIMULA were included in Version 7.00 of MVS SIMULA and this system was installed shortly after Version 6.02 of CMS SIMULA expired in August 1979. This new system repaired some of the problems with earlier versions but there was another internal error which was subsequently corrected by the Norwegian Computing Center.

A significant fragment of the actual code for the APL implementation appears in the manuscript of a monograph prepared as part of the work of this grant and, therefore, a suitable publication format of the program text was required. IBM SIMULA systems before Version 7.0 were upper case only and to provide publication format program text a program to format SIMULA programs was imported from DEC-10 SIMULA and modified for use in the CMS environment. Until August 1979, when Version 7.0 was first available, two copies of program text were maintained: an executable copy and a publication copy. When Version 7.0 of IBM SIMULA became available, the execution text was destroyed because the publication format text became executable. This change in the SIMULA compiler made it significantly easier to work with the system.

The work on file system development summarized in Section 4 provides both the terminal interface and other support for the APL implementation and is an essential part of the work on the APL implementation.

A clean proof of the correctness of the APL implementation up to the evaluation of expressions has been completed and is part of a draft research monograph. This includes the correctness of the implementation of APL individuals, primitive functions, symbol tables and an expression evaluator. The formal semantics of user-defined functions and operators has been completed and has been implemented. The essential arguments in the proof of correctness of this part of the implementation are recorded in rough draft form.

3. Verification of APL Programs

Bringing the interactive verifier that was implemented in DEC-10 SIMULA into reliable and useable operation in IBM SIMULA was a long and difficult process and adequately reliable operation was first achieved in January 1980. The difficulties described above for the APL implementation caused more serious problems when working with the verifier.

A genuinely interactive verifier makes heavy demands on the file management capabilities of an operating system and it was quickly discovered that an enhanced file system would be needed to provide adequate facilities while working with the verifier. The work on a file system described in Section 4 provides essential support for the verifier in its present form and some of the capabilities that are needed to fully support work with APL programs.

The main problem that was encountered with the verifier is related to a number of serious errors in the SIMULA compiler/run time system. This program makes very heavy demands on the storage manager, the procedure calling mechanism and the coroutine linkage mechanism and the implementation of these aspects of SIMULA is particularly weak in IBM SIMULA. The Norwegian Computing Center has devoted substantial efforts to solving these problems and each version of IBM SIMULA has new code for this part of the system. At each step in this improvement, small additional problems and incompatibilities arise.

Each version change of the SIMULA system required careful checking of the program and subtle modifications to avoid the current version of bugs in the system. It has been observed that code which was introduced to cause the program to execute more reliably in one version caused problems in a later version.

Version 7.00 of MVS SIMULA is much more reliable than earlier versions but the run time system caused unpredictable terminations for protection or addressing exceptions. A fortunate combination of events led to a useful solution of these problems.

The University Computing Center changed from VM/CMS 5.8 to VM/CMS 6.5 and at the same time Version 7.00 of CMS SIMULA became available. With the active support of Computing Center staff members, it was determined that most of the run time system problems can be avoided by using the CMS SIMULA compiler with the MVS SIMULA run time system under VM/CMS 6.5. Although this strange combination of software is not without problems, it is the most reliable working environment that is available and was adequate. This combination was identified in January 1980.

After the imported verifier came into reliable operation in the VM/CMS environment, the program was divided into several logically related segments of code and this code now forms the basis for a laboratory to construct a variety of programs that are useful both for this research and in other research and instructional activities.

In addition, the verifier was modified so that those aspects of the verifier which depend on the syntax and semantics of the language in which programs are to be verified are contained in separate SIMULA classes. The program was modified to read both syntactical and semantic definitions from input files. In order to provide a verifier for APL expressions, an input grammar and a set of semantic rules are required. This work was abandoned in order to complete the verified APL implementation discussed in Section 2.

There is no checkpoint facility available in IBM SIMULA and it is quite often desirable to be able to interrupt a terminal session with the verifier and resume at a later time. In order to do this, the file system that supports the verifier was equipped with the capability to write log files and the capability to read indirect files while reproducing a terminal transcript on a terminal while an input file is being read. This is described in more detail in Section 4.

Work on the verifier consumed a significant fraction of the human and computing resources of the grant from the beginning of the grant until the end of April 1980 when attention was restricted to the APL implementation. After this, work on APL consumed essentially all human and computing resources available. While it is not possible to give exact figures, work on the verifier probably consumed 65% of the computing resources and 60% of the human resources available for this work. This assumes a division of work on the file system between the two major programs. If work on the verifier had continued, a much larger fragment of the computing resources would have been used on the verifier.

4. File System Development

Since the principal objective of this research program is the verification of APL programs, work on the design and implementation of an input/output system for IBM SIMULA requires some explanation.

Both the incremental verifier and the APL implementation which are an important part of this research are strongly interactive programs and these programs require dynamic file naming (that is, the ability to name and open a file during execution) if they are to function in an acceptable manner. IBM SIMULA as provided by the Norwegian Computing Center and CMS SIMULA as provided by Imperial College use OS/370 simulation macros to perform input/output. This system suffers from many limitations when working with interactive programs:

The CMS simulation of OS/370 input/output requires that file definitions be provided to a program before execution begins. Moreover, once a file definition has been executed, the file is known to a program by a DD name rather than the file name. When working with both the verifier and the APL implementation, one does not, in general, know which files will be read or written during execution. One possible solution to this problem is to issue a large number of file definitions before program execution and then retain a written record of the correspondence between DD names and file names. Even with the help of the CMS EXEC processor, this was found to be very time consuming and the source of many errors particularly when a CRT terminal is used.

Terminal support for program input/output in the IBM SIMULA implementation is a byproduct of this use of OS simulation macros. Output written to the terminal is double buffered and, therefore, extensive program modifications were needed to write extra blank lines to make sure that a prompt is printed before the response is read from the terminal. An empty line of input is treated as an end-of-file and an accidental strike of the return key

causes the termination of execution with a loss of all work in the current execution of a program.

In early work with CMS, *ad hoc* solutions to these problems were employed and it became quite clear that these problems would persist and become more and more difficult. Therefore, a comprehensive solution to these and other problems was undertaken.

The first approach to the solution of this problem was the design and implementation of a SIMULA class DIALOG which contains procedures to provide dynamic file naming and a minimal set of primitives for terminal dialog. In addition, a number of procedures which are available in the DEC-10 SIMULA library but which are unavailable in the IBM SIMULA library were added to this class.

The first version of DIALOG greatly simplified work with the verifier but it quickly became obvious that there were still too many possibilities for errors. In addition, it was still impossible to reproduce the prompt by indentation that is used in APL and to provide some prompts from the verifier in a reasonable way.

A second version of DIALOG was constructed to provide additional security and prevent unexpected termination of execution because of some trivial error. In addition, assembly code was written to provide the output file attribute *Breakoutimage* which is available in DEC-10 SIMULA but unavailable in IBM SIMULA. This class appeared to solve many of the problems that were encountered.

The two versions of DIALOG are described in technical memoranda 79-3 and 79-3a; the former is no longer available because it is obsolete. As work with this class continued, two problems became quite obvious:

The declaration of class DIALOG is approximately 2500 lines of SIMULA code and is supported by about 300 lines of assembly code. This imposes a significant overhead when compiling the verifier or the APL implementation. A single compile was costing \$20. [Although IBM SIMULA supports a restricted form of separate compilation of classes, the computing resources required when separate compilations are used is greater than the resources required for a complete compilation.]

Both the APL verifier and the APL implementation must interact with terminals and files written in the standard character set (ASCII or EBCDIC) as well as the APL character set. There are three APL character sets: key-paired, bit-paired and tty-code. Only the first two are of interest for this work. Character set translation was required so that the running program would be unaware of the character set of the input/output device. This translation should be performed by the input/output system and it was quite obvious that an extension of DIALOG to provide these capabilities would be both inefficient and quite unreliable because it would extend DIALOG far beyond what was anticipated in the original design.

Therefore, a comprehensive design of a file system that is a proper extension of the SIMULA Common Base Definition was undertaken. The view of files adopted in this design is quite different from the standard CMS view and is much easier to use.

The first assumption is that a file is named only by the name of the file as it appears in the CMS directory. Moreover, simply writing to a file after giving it a name is sufficient to cause the file to exist. Running programs may dynamically reference both input and output files using only the name of the file. If a program is to read or write different files in different executions, the file names are read from the terminal.

The second assumption is that a file may be associated with any input or output device and from the vantage point of a running program there is no difference between devices except for the name of the file. The various file formats that are used by IBM systems are of no concern to the program or to the author of the program.

The third assumption is that the data in a file may be written in one of three character sets: EBCDIC, key-paired APL or bit-paired APL. Further, a program may interpret characters coming from a file or written to a file as either a sequence of EBCDIC characters or as a sequence of key-paired APL characters. It is the responsibility of the file system to provide appropriate translations in accord with specifications given when the file is opened.

A design requirement is that the files actually read and written must be standard CMS files so that programs written using this file system can communicate with other programs by way of the CMS file system.

A second design requirement is that all run time errors must be detected and reported to the user with an opportunity for corrective action from the terminal rather than a simple error termination.

The design of this file system is described in Technical Memorandum 79-8a and a prototype of this file system was implemented by modifying much of the code that was originally part of class DIALOG.

This prototype file system solved many problems and made the software development work of the grant much easier. It was also exported to a number of other CMS SIMULA users.

The prototype system was used to construct a stream input class that supports indirect files for use in the incremental verifier. When this code was used in production, a number of strange things began to happen -- parts of files were omitted in some executions but not in others. After a more careful investigation it was discovered that this was a result of timing problems in the operating system.

In the prototype system, terminal input/output is performed by assembly coded procedures that use the CMS input/output system and file input/output is performed using OS simulation macros as used by the SIMULA run time system. System documentation for CMS indicates that this is an acceptable combination but a number of CMS installations report timing problems similar to the ones encountered in this work.

To avoid these problems, an assembly coded version of input/output procedures for disk files and virtual readers, printers and punches was designed and may have been implemented. The results of this work are not available and it is not possible to give a precise reason. Two possible explanations are supported by available information and it is not possible to decide which is correct.

Just after the programmer that was working on this implementation reported that the work was completed and ready for testing, the CMS mini-disks assigned to the programmer were erased. The University Computing Center restored these mini-disks from tapes that were written after the programmer reported that work was completed but before the principal investigator tested the programs. The programs in question were not on the tape. There are two possible explanations for this state of affairs:

(1) The computing center did not restore the mini-disks from the correct tape. The computing center has records to support the claim that the correct tape was used but this

evidence is not conclusive. (2) The work was not completed by the programmer and the disks were erased to hide the fact that the work was not completed. Examining the date-time stamps of the restored files and accounting records support this hypothesis but the available evidence is not conclusive.

The programmer that was doing this work accepted another position with the Virginia Tech Department of Computer Science before the termination of this grant and when this problem was brought to the attention of the Head of the Computer Science Department he directed the programmer to again complete this work. As of this date, the work is incomplete and there is no reason to expect that it will be completed.

This problem arose largely as a consequence of the fact that the programmer was working at the Virginia Tech main campus in Blacksburg, approximately 300 miles from the location of the principal investigator. This made direct supervision of the individual's work difficult.

5. Other Software Development

The incremental verifier parses programs and assertions using an SLR(1) parser. The parser in the verifier reads the state table of the parser and a list of the tokens of the language from an input file and, therefore, is independent of the grammar of the language to be parsed.

Such parse table files are produced from a BNF grammar by an SLR(1) parser generator that was written in DEC-10 RATFOR at the University of Arizona by Dr. F. C. Druseikis. Both the RATFOR and Fortran source files for this program were transferred to Virginia Tech. This program consists of approximately 4000 lines of RATFOR source code. While the Fortran code is quite acceptable to the DEC-10 Fortran compiler, it is completely unacceptable to the IBM Fortran compiler. In addition, the parser generator relies on the representation of integers on the host hardware. This representation is a parameter in the RATFOR code but appears as many constants in the corresponding Fortran code. To simplify the transfer of this program as well as some small utilities and to provide a generally useful tool for program development at Virginia Tech, a RATFOR preprocessor that is the same as the UNIX and TOPS-10 version but which writes Fortran that is accepted by the IBM compiler was constructed using part of the code from this RATFOR implementation.

This RATFOR preprocessor is now in general use at Virginia Tech and has been exported to other CMS installations.

The RATFOR preprocessor was used to bring the SLR(1) parser generator into reliable operation in the VM/CMS environment.

A variety of utility programs to deal with problems that are related to the physical location of the Principal Investigator and to limitations in VM/CMS system utilities were constructed. These include programs to correct different translations from ASCII to EBCDIC, to simulate an upper and lower case line printer on a 30 cps hardcopy terminal, to perform line printer spooling of files that contain tab characters, etc.

After the Principal Investigator decided to leave Virginia Tech it became clear that it was very desirable to record the software that was produced under this grant on magnetic tape in a form that can be read by a variety of computing machines. The university computing

center provides only a limited utility for writing files on tape and the tape format written by this utility caused serious problems when attempting to read such tapes on a CMS system and the format is not suitable for reading on other systems. Therefore, a tape dump and restore utility that can be used to read and write tapes for use with other computing systems was written. This task was made substantially more difficult by a computing center policy that prohibits the use of tapes on the VM/CMS system. It was necessary to convert files into card images so that tape write jobs could be transmitted to an MVS system running on another processor. This also required code to reconstruct files written in this way when tapes are read.

6. Publications

A manuscript, "Files in an Interactive Environment", describing some of the work on file systems, has been submitted for publication in *Software Practice & Experience*; referees reports are awaited.

A partial draft of a research monograph, "A Primitive Recursive Semantics and Implementation of APL", has been submitted for publication as a volume in the Springer-Verlag series *Lectures in Computer Science*. The first referee strongly praised the content of the manuscript and suggested that it be reviewed by a second referee and that the text be revised to make it more pleasant to read.

Professor Juris Hartmanis, editor of this series, had indicated that he is favorably disposed toward publishing the monograph and the Principal Investigator has indicated that he is prepared to make significant changes in the text for final publication.

The typescript for the draft manuscript was prepared using a conventional typewriter and is not in machine readable form. The program text of the APL implementation that is discussed in this manuscript has been transferred to disks at Xerox and is available for use with the word processing system that was used to create this document. Xerox has promised to provide some support for the preparation of the final monograph and the Principal Investigator is prepared to continue this work as soon as it is accepted for publication.

A total of seventeen Technical Memoranda that describe work done under this grant have been written and distributed. Of the seventeen, fifteen are still available and the other two are obsolete.

Much of the supporting software for this research has been exported to other CMS installations and is being used actively.

7. Personnel

Dr. Richard J. Orgass directed his efforts toward bringing the software that is used in this research into operation in the VM/CMS environment, toward completing a formal definition of the syntax and semantics of APL, toward completing a verified implementation of APL and toward constructing a verifier for APL programs.

Final Scientific Report

To solve the file management problems, he designed and implemented the SIMULA class DIALOG for the CMS environment and designed and partially implemented an interactive file system. This work on the file system is documented in technical memoranda.

He spent a significant amount of time working with the Norwegian Computing Center to solve problems related to bringing the verifier into operation and to constructing a usable working environment out of parts of the verifier and other software.

Dr. J. J. Martin, Associate Professor of Computer Science at Virginia Tech, devoted one month to the work of the grant. During this time he familiarized himself with the verification techniques and programs that are used in this work. He has written a technical report describing many of the capabilities of the SLR(1) parser which was used in other parts of the research program.

These faculty members were assisted by a number of student employees of the grant; their activities are summarized below.

Mr. Stephen M. Choquette implemented the CMS RATFOR system with general guidance from Dr. Orgass. Mr. Choquette received a B.S. in Computer Science from Virginia Tech in June 1979 and is now employed by IBM and a part time student in the Northern Virginia Graduate Program of Virginia Tech. His full time position and studies leave no further time for work on this grant.

Mr. Robert Porter was employed as a Graduate Research Assistant during the summer of 1979. He assisted Dr. Orgass in the implementation of DIALOG and supported the work of Dr. Martin.

Mr. R. D. Johnson, a sophomore at Virginia Tech, worked for the grant intermittently during the 1979-1980 academic year. He assisted Dr. Orgass in the implementation of the interactive file system by writing the assembly code that supports this file system.

Mr. J. W. Stibbards, a Virginia Tech senior, was briefly involved in the work of the grant but found that he had too many other commitments and withdrew from this work. This was a wise decision because his educational program is far more important.

Mr. R. Critz, a Virginia Tech sophomore taking a year off from his formal education, worked for the grant as a programmer. He brought the SLR(1) parser generator into operation and has written several assembly procedures to facilitate the management of the CMS environment. He also served the grant as program librarian and kept the grant software in order. He was to have implemented an assembly coded file system that only used the CMS input/output macros but this work is not available.

A continuing problem for this grant was the lack of graduate assistants to work on the research program. All of the graduate students in the Northern Virginia Program have full time jobs and their commitments to work, studies and family preclude any other activities. University regulations prohibit paying full time salaries that might attract a qualified programmer in the Metropolitan Washington area and it was not possible to use a full time programmer in place of graduate students.

8. Publications

A list of Technical Memoranda describing work performed under this grant follows. Copies of reports numbered 79-1 to 80-1 were transmitted to AFOSR as attachments to 80-2. Copies of reports numbered 80-3 to 80-6 are attachments to this original of this report.

- 79-1 R. J. Orgass. *Concerning Classes Within Classes*.
January 15, 1979.
- 79-2 R. J. Orgass. *Line Printer Spooling on an ASCII terminal*.
April 12, 1979.
- 79-3a R. J. Orgass. *A SIMULA Class for Writing Interactive Programs*. October 14, 1979.
- 79-4 S. M. Choquette and R. J. Orgass. *CMS RATFOR System Manual*.
July 1, 1979.
- 79-5 S. M. Choquette and R. J. Orgass. *CMS RATFOR User's Manual*.
July 1, 1979.
- 79-6 R. J. Orgass. *CMS Software Notebook (First Edition)*.
July 31, 1979.
- 79-7 R. J. Orgass. *Converting DEC-10 SIMULA Programs to CMS SIMULA*.
August 3, 1979.
- 79-8a R. J. Orgass. *Design for a CMS SIMULA File System with Four Character Sets*.
November 5, 1979.
- 80-1 J. J. Martin. *A SIMULA Program for SLR(1) Parsing*.
January 1980.
- 80-2 R. J. Orgass. *Interim Scientific Report, AFOSR Grant 79-0021*.
February 15, 1980.
- 80-3 R. J. Orgass. *Files in an Interactive Environment*.
April 1, 1980.
- 80-4 R. J. Orgass. *LIBSIM: An Extension of the SIMULA Library*.
August 28, 1980.
- 80-5 R. J. Orgass. *Tape Dump and Restore*.
August 28, 1980.
- 80-6 R. J. Orgass. *SIMULA File Classes*.
October 8, 1980.